

TIDE 2.0: A Model-Agnostic, Utility-Preserving Open Source Tool for Distributed Clinical Text De-identification

Jose Posada, PhD; Somalee Datta, PhD; Priya Desai, MSc

Technology and Digital Solutions, Stanford Health Care, Stanford, CA, USA

Background

Clinical free text contains critical observations such as nuanced symptoms, treatment responses, and pathology details that are usually absent from structured electronic health record (EHR) fields. Within the OHDSI community, these narratives are ingested via ETL into the OMOP Common Data Model (CDM) NOTE table [1], serving as the foundation for traditional (e.g., keyword-based cohort selection) and modern, AI-powered observational research (e.g., LLM-based clinical phenotyping). However, deeply embedded protected health information (PHI) makes clinical notes highly challenging to share and use safely, typically requiring de-identification before research use.

Under the US HIPAA Safe Harbor, de-identification requires removing 18 identifier categories [2]. However, simple redaction (e.g., replacing names and dates with generic placeholders like [REDACTED]) destroys the longitudinal coherence of the text. Clinical observational research relies on contextual integrity, such as tracking consistent patient and provider references across multi-year encounters and preserving chronological order. If timelines are broken by random date shifting or blanking, longitudinal analyses (e.g., computing drug-exposure-to-outcome intervals) become impossible.

Existing open-source tools only partially address these needs. Frameworks like Microsoft Presidio offer flexible recognizers but lack healthcare-specific models and an execution layer for millions of notes [3]. Our prior toolkit, TIDE 1.0, used a Java pipeline with Stanford CoreNLP and Apache Beam [4]. While functional, its NER model is outdated, its recognition and anonymization stages were tightly coupled, and it lacked robust fault tolerance or utility-preserving longitudinal anonymization.

Rather than another standalone model, the OHDSI community needs a reproducible, model-agnostic pipeline that transforms any state-of-the-art recognizer into an enterprise-scale, utility-preserving engine for local execution. TIDE 2.0 fills this gap, enabling sites to safely de-identify narratives within their OMOP-CDM ETL workflows and run consistent, high-throughput phenotyping across distributed research networks.

Methods

TIDE 2.0 is an open-source (MIT-licensed) Python 3.12 library (tide2-core, <https://github.com/susom/tide2-core>) that processes raw clinical documents into de-identified notes and structured PHI-span metadata for OMOP-CDM ETL workflows. Its central architecture cleanly

decouples two stages: a Recognizer Stage to detect PHI spans, and an Anonymizer Stage to replace them. This separation allows sites to run, audit, cache, and re-run recognition without re-anonymizing, facilitating easy strategy swapping at either stage.

Recognition is model-agnostic; any pretrained Hugging Face NER token classifier can be registered via a Presidio-compatible interface without pipeline changes [3]. For on-premise execution, TIDE 2.0 provides tide2-sentry (<https://huggingface.co/stanford-med-hdr/tide2-sentry-clinical-ner>), a distilled small language model trained on the SHIELD benchmark [7]. To support modern clinical AI, TIDE 2.0 also includes an LLM-recognizer class that prompts generative models to extract PHI spans as structured JSON. This interface integrates fast regular-expression and Aho–Corasick matchers, identifying structured identifiers (e.g., local medical record numbers, encounter IDs, SSNs) 10–100× faster than deep learning models.

Anonymization preserves longitudinal utility using the "hidden-in-plain-sight" method [5]. Instead of deleting identifiers, TIDE 2.0 employs cryptographic, deterministic replacement: a given name, medical record number, or provider reference maps consistently to the same synthetic surrogate across a patient's entire record. Date jittering is also deterministically derived per patient, safely shifting the timeline while preserving precise intervals between medical events for complex temporal analyses.

To scale, TIDE 2.0 leverages the Ray distributed execution framework [6] in a two-stage GPU/CPU pipeline: GPU actors handle high-throughput model inference, while concurrent CPU actors manage document chunking, reassembly, regex matching, and anonymization. The pipeline features adaptive GPU batching and automatic batch splitting in response to out-of-memory (OOM) errors, ensuring robust fault tolerance for enterprise jobs. The library includes two CLI tools: tide2-runner for execution and tide2-visualizer (a Streamlit app) for side-by-side verification.

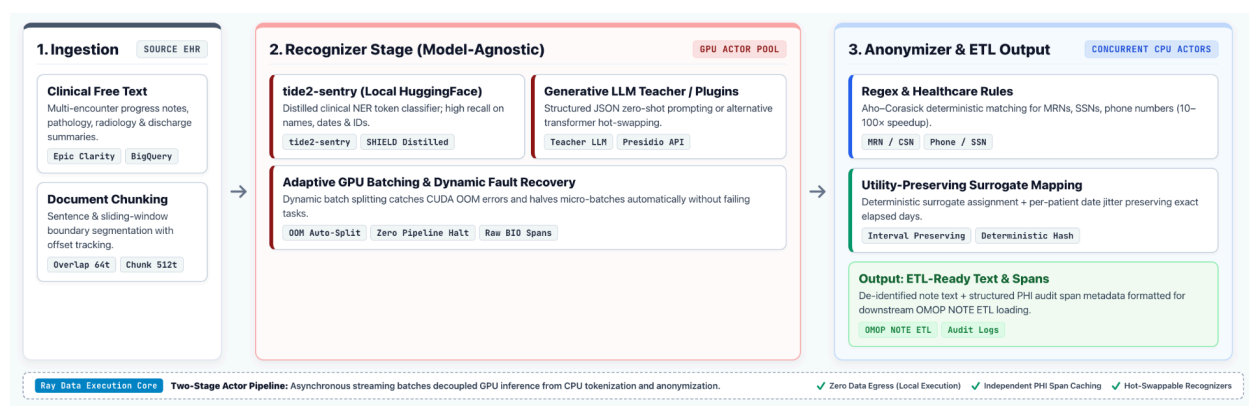


Figure 1. TIDE 2.0 architecture and execution flow. Ray orchestrates concurrent CPU and GPU actor pools with active fault tolerance. Decoupled stages allow sites to audit PHI spans or swap models without changing downstream database integration or anonymization logic.

Results

Figure 1 outlines the TIDE 2.0 architecture and CPU/GPU data flow. This figure demonstrates the TIDE 2.0 pipeline's architectural flexibility, model swapping, and preservation of the clinical utility of the OMOP NOTE table.

1. Model-Swapping and Distillation Fidelity (tide2-sentry)

The decoupled architecture allows sites to swap underlying recognizers to match specific workloads easily. For traditional cohort research, sites can deploy ultra-fast, CPU-bound regex and Aho-Corasick matchers to strip standard identifiers with minimal compute. For AI-powered research, sites can hot-swap deep-learning models or generative LLMs without altering the ingestion pipeline. We validated distillation fidelity on the multi-site SHIELD dataset¹ [7] by comparing the LLM teacher against the compact, on-premise student model tide2-sentry². Figure 2 illustrates Precision and Recall differences from teacher to student.

2. Impact on Anonymization and Downstream Clinical Utility

Traditional tools destroy clinical utility by blanking identifiers, rendering narratives useless for longitudinal research. TIDE 2.0 preserves semantic flow and chronological spacing (Figure 3), directly benefiting downstream analyses:

- **Timeline Preservation:** Shifting dates deterministically (e.g., maintaining an exact 32-day interval between admission and follow-up) preserves temporal intervals for survival analysis and modeling.
- **Consistent Mapping:** Mapping "John Smith" to "Rowe Keegan" and MRN "1234567" to "2876742" across all notes keeps multi-note patient records linked.
- **Syntactic Naturalness:** Replacing PHI with realistic surrogates instead of blanking (e.g., [REDACTED]) maintains syntactic integrity, crucial for the accuracy of downstream NLP and generative models.

3. Comparison with TIDE 1.0

To demonstrate how TIDE 2.0 enables modern research pipelines, Table 1 compares the core capabilities with our previous Java version, TIDE 1.0.

¹ (<https://stanford.redivis.com/datasets/ee4n-6gv2f84et>)

² (<https://huggingface.co/stanford-med-hdr/tide2-sentry-clinical-ner>)

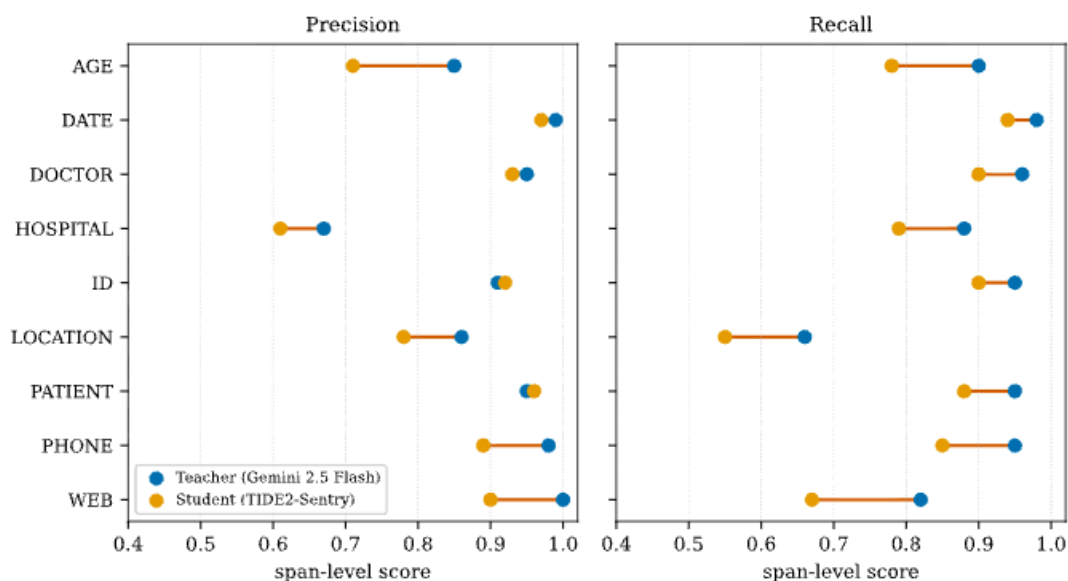


Figure 2. Distillation fidelity dumbbell plot across SHIELD categories comparing the LLM teacher against the compact student model tide2-sentry.

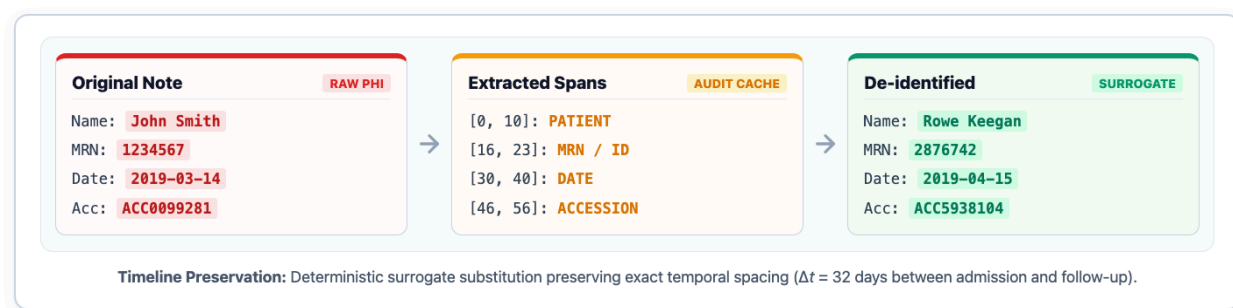


Figure 3. Example of TIDE 2.0 deterministic and utility-preserving substitution vs. naive redaction.

Conclusions

TIDE 2.0 significantly advances our prior Java-based pipeline [4] by using a Python-based, Ray-distributed architecture that provides clinical sites with an enterprise-scale, fault-tolerant de-identification tool to prepare clinical narratives for ETL ingestion into OMOP-CDM NOTE tables.

For the OHDSI community, TIDE 2.0 demonstrates that protecting patient privacy does not require sacrificing utility or committing to static models. Its model-agnostic recognizer allows sites to dynamically adopt rule-based, transformer, or LLM models (such as tide2-sentry), while its interval-preserving anonymizer keeps clinical narratives viable for temporal observational research. We invite the community to adopt and contribute to this open-source tool.

Dimension	TIDE 1.0 [4]	TIDE 2.0
Stack	Java 8 / Apache Beam	Python 3.12 / Ray Data
Architecture	Monolithic (tightly coupled)	Decoupled (Recognizer & Anonymizer stages)
Models	Static Stanford CoreNLP	Model-agnostic (Regex, Hugging Face tide2-sentry, LLM)
Fault Tolerance	Entire job restart	Dynamic GPU batch splitting on CUDA OOM
Utility Preservation	Naive blanking ([REDACTED])	Deterministic surrogate mapping & per-patient date jitter
OMOP Integration	Manual file exports	Native OMOP CDM NOTE table ETL & PHI audit logging

Table 1. Functional and architectural comparison between TIDE 1.0 and TIDE 2.0.

References

1. Overhage JM, Ryan PB, Reich CG, Hartzema AG, Stang PE. Validation of a common data model for active safety surveillance research. *J Am Med Inform Assoc.* 2012;19(1):54-60. doi:10.1136/amiajnl-2011-000376
2. Office for Civil Rights, U.S. Department of Health and Human Services. Guidance regarding methods for de-identification of protected health information in accordance with the HIPAA Privacy Rule. 2012. <https://www.hhs.gov/hipaa/for-professionals/privacy/special-topics/de-identification/>
3. Microsoft. Presidio: data protection and de-identification SDK. <https://github.com/microsoft/presidio>
4. Stanford Center for Population Health Sciences. TIDE 1.0: Text Iterative De-identification. <https://github.com/susom/tide>
5. Carrell D, Malin B, Aberdeen J, Bayer S, Clark C, Wellner B, Hirschman L. Hiding in plain sight: use of realistic surrogates to reduce exposure of protected health information in clinical text. *J Am Med Inform Assoc.* 2013 Mar-Apr;20(2):342-8. doi:10.1136/amiajnl-2012-001034
6. Ray Data: Scalable Data Processing for AI Workloads. <https://docs.ray.io/en/latest/data/data.html>
7. Posada JD, Love D, Datta S, Desai P. SHIELD: A Diverse Clinical Note Dataset and Distilled Small Language Models for Enterprise-Scale De-identification. *arXiv preprint arXiv:2605.03301.* 2026. <https://arxiv.org/abs/2605.03301>

8. Observational Health Data Sciences and Informatics (OHDSI). OMOP Common Data Model v5.4.
<https://ohdsi.github.io/CommonDataModel/cdm54.html>